APPENDIX A

Windows NT. Server

Windows Home Pages | Downloads | Support

Enter a search phrase:

[Search]

Advanced Search

Windows NT Server Home Page

Product Information

Technical Resources

Downloads

Support

Related Sites

Partners

Maintain Security with Windows NT Server 4.0

Support Contents

Product Availability and FAQ

Top Issues and FAQs

Contact Support

Windows NT Server Newsgroups

Microsoft Cluster Server General Questions

- **Cluster Basics**
- **Intro to Microsoft Cluster Server**
- **High Availability**
- **Manageability**
- **Scalability**
- **Application and Service Support**

Cluster Basics   ▲ top of page

What is a server "cluster"?
A server cluster is a group of independent servers managed as a single system for higher availability, manageability, and scalability.

What does it take to create a server cluster?
The minimum requirements for a server cluster are (a) two servers connected by a network, (b) a method for each server to access the other's disk data, and (c) special cluster software like Microsoft Cluster Server (MSCS). The special software provides services such as failure detection, recovery, and the ability to manage the servers as a single system.

What are the features of server clustering?
There are three primary features to server clustering: availability, manageability, and scalability. Using Microsoft Cluster Server as an example:

- *Availability*: MSCS can automatically detect the failure of an application or server, and quickly restart it on a surviving server. Users only experience a momentary pause in service.
- *Manageability* : MSCS lets administrators quickly inspect the status of all cluster resources, and move workload around onto different servers within the cluster. This is useful for manual load balancing, and to perform "rolling updates" on the servers without taking important data and applications offline.
- *Scalability*: "Cluster-aware" applications can use the MSCS services through the MSCS Application Programming Interface (API) to do dynamic load balancing and scale across multiple servers within a cluster.

What are clusters used for?
Customer surveys indicate that MSCS clusters are used as highly available multipurpose platforms, mirroring the current uses of the Microsoft Windows NT Server operating system. Surveyed customers suggested that the most

common uses of MSCS clusters are mission-critical database management, file/intranet data sharing, messaging, and general business applications.

When a cluster is recovering from a server failure, how does the surviving server get access to the failed server's disk data?
There are basically three techniques that clusters use to make disk data available to more than one server:

- *Shared disks*: The earliest server clusters permitted every server to access every disk. This originally required expensive cabling and switches, plus specialized software and applications. (The specialized software that mediates access to shared disks is generally called a Distributed Lock Manager, or DLM.) Today, standards like SCSI have eliminated the requirement for expensive cabling and switches. However, shared-disk clustering still requires specially modified applications. This means it is not broadly useful for the wide variety of applications deployed on the millions of servers sold each year. Shared-disk clustering also has inherent limits on scalability since DLM contention grows geometrically as you add servers to the cluster. Examples of shared-disk clustering solutions include Digital VAX Clusters and Oracle Parallel Server.
- *Mirrored disks* : A flexible alternative is to let each server have its own disks, and to run software that "mirrors" every write from one server to a copy of the data on at least one other server. This is a great technique for keeping data at a disaster recovery site in synch with a primary server. A large number of disk mirroring solutions are available today; examples for the Windows NT Server environment are available from Network Specialists (NSI), Octopus, Veritas, and Vinca. Many of these mirroring vendors also offer cluster-like high-availability extensions that can switch workload over to a different server using a mirrored copy of data. However, mirrored-disk failover solutions cannot deliver the scalability benefits of clusters. It is also arguable that they can never deliver as high a level of availability and manageability as shared-disk clustering since there is always a finite amount of time during the mirroring operation in which the data at both servers is not 100 percent identical.
- *"Shared nothing"*: In response to the limitations of shared-disk clustering, modern cluster solutions employ a "shared nothing" architecture in which each server owns its own disk resources (that is, they share "nothing" at any point in time). In the event of a server failure, a shared-nothing cluster has software that can transfer ownership of a disk from one server to another. This provides the same high level of availability as shared-disk clusters, and potentially higher scalability since it does not have the inherent bottleneck of a DLM. Best of all, it works with standard applications since there's no special disk access requirements. Examples of shared-nothing clustering solutions include Tandem NonStop, Informix Online/XPS, and Microsoft Cluster Server.

Intro to Microsoft Cluster Server　　　　　　　　　▲ top of page

What is "Wolfpack"?
"Wolfpack" was the code name for Microsoft Cluster Server.

What is Microsoft Cluster Server (MSCS)?
MSCS is a built-in feature of Windows NT Server, Enterprise Edition. It is software that supports the connection of two servers into a "cluster" for higher availability and easier manageability of data and applications. MSCS can automatically detect and recover from server or application failures. It

can be used to move server workload to balance utilization and to provide for planned maintenance without downtime. And, over time, MSCS will also become a platform for highly scalable, cluster-aware applications.

How many servers can be in an MSCS cluster?
The initial release of MSCS will be supported on clusters with two servers. A future version referred to as MSCS "Phase 2" will support larger clusters, and will include enhanced services to simplify the creation of highly scalable, cluster-aware applications.

What other companies were involved in the development of MSCS?
Microsoft worked closely with leading hardware vendors, software vendors, and customers in the specification and development of MSCS and its API. These other companies participated through five different programs:

- *Strategic alliances*: Microsoft formed strategic alliances with two of the key pioneers in clustering technology: Digital Equipment Corporation (in 1995) and Tandem Computers (in 1996). In both of these alliances, patent portfolios were cross-licensed, and Microsoft gained access to proven clustering expertise and technology, plus a strong partner committed to helping extend that technology to benefit customers of Windows NT Server.
- *Early Adopter vendors*: Starting with the announcement of the MSCS project in October 1995 and extending through the beta test program, Microsoft worked closely with six leading system vendors who provided support, expertise, and sample cluster configurations to support the development of MSCS. The Early Adopter system vendors were Compaq Computer Corporation, Digital Equipment, Hewlett-Packard, IBM, NCR, and Tandem Computers.
- *Open Process*: Whenever Microsoft extends the Microsoft Win32® API, as it did with MSCS, it enlists the participation of vendors and customers in its "Open Process." This is a series of confidential design previews and specification reviews that assures that the resulting API is robust, complete, and usable by a broad segment of the industry. More than 60 organizations participated in the MSCS Open Process sessions, which took place between January and July of 1996.
- *SDK previews*: Microsoft first provided early copies of the MSCS Software Development Kit to the 60+ Open Process organizations in September of 1996, and distributed a more advanced preview SDK to more than 2,000 developers at the November 1996 Microsoft Professional Developers Conference.
- *Beta test program*: MSCS Beta 1 was shipped in December 1996 to 350 customer and vendor sites. Beta 2 shipped in April 1997 to more than 750 sites. And Beta 3 of MSCS was shipped as an embedded feature of Windows NT Server, Enterprise Edition 4.0 Beta 2 in July 1997 to more than 2,100 sites. Each of these betas was also available to thousands of additional developers and customers through the Microsoft Developers Network (MSDN) Level III.

In what languages will MSCS be available?
Microsoft Windows NT Server, Enterprise Edition 4.0, which included MSCS 1.0, will be offered in English, French, German, Japanese, and Spanish.

Through what channels will Windows NT Server, Enterprise Edition be available?

PAGE 19/28 * RCVD AT 2/13/2004 3:00:32 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/1 * DNIS:8729306 * CSID:215 * DURATION (mm-ss):08-10

2/11/200

Microsoft Windows NT Server, Enterprise Edition will be available to
customers through all standard channels: reseller, retail, OEM, and the
Microsoft Select licensing program.

What versions of Windows NT Server will MSCS support?
MSCS software will only be available as a built-in feature of Windows NT
Server, Enterprise Edition.

**Will MSCS be extended beyond Windows NT Server to Windows NT
Workstation?**
There is currently no plan to extend cluster support to Windows NT
Workstation. MSCS software has been designed and written to closely
integrate with the architecture and features of Windows NT Server, including
its server-oriented networking and directory services capabilities.

What clients can connect to an MSCS cluster?
Any client that can connect to Windows NT Server through TCP/IP will work
with MSCS. This includes Microsoft MS-DOS, Microsoft Windows 3.*x*, Windows
95, Windows NT, Apple Macintosh, and UNIX. MSCS does not require any
special software on the client for transparent recovery of services that
connect to clients through standard IP protocols.

High Availability   ▲ top of page

How does MSCS provide high availability?
MSCS uses software "heartbeats" to detect failed applications or servers. In
the event of a server failure, it employs a "shared nothing" clustering
architecture that automatically transfers ownership of resources (such as disk
drives and IP addresses) from a failed server to a surviving server. It then
restarts the failed server's workload on the surviving server. All of this--from
detection to restart--typically takes under a minute. If an individual
application fails (but the server does not), MSCS will typically try to restart
the application on the same server; if that fails, it moves the application's
resources and restarts it on the other server. The cluster administrator can
use a graphical console to set various recovery policies, such as
dependencies between applications, whether or not to restart an application
on the same server, and whether or not to automatically
"failback" (rebalance) workloads when a failed server comes back online.

Can MSCS provide "zero downtime"?
No. MSCS can dramatically reduce planned and unplanned downtime.
However, even with MSCS, a server could still experience downtime from the
following events:

- *MSCS failover time*: If MSCS recovers from a server or application failure,
  or if it is used to move applications from one server to another, the
  application(s) will be unavailable for a non-zero period of time (typically
  under a minute.)

- *Failures which MSCS can't recover*: There are types of failure that MSCS
  does not protect against, such as loss of a disk not protected by RAID,
  loss of power when a UPS isn't used, or loss of a site when there's no

fast-recovery disaster recovery plan, but most of these can be survived
with minimal downtime if precautions are taken in advance.
- *Server maintenance that requires downtime*: MSCS can keep applications
  and data online through many types of server maintenance, but not all
  (for example: completely upgrading both servers in a cluster, or installing
  a new version of an application which has a new on-disk data format that
  requires reformatting preexisting data).

Microsoft recommends that clusters be used as one element in customers'
overall programs to provide high integrity and high availability for their
mission-critical server-based data and applications.

Is MSCS failover transparent to users?

MSCS does not require any special software on client computers, so the user
experience during failover depends on the nature of the client side of their
client-server application. Client reconnection is often transparent, because
MSCS has restarted the applications, file shares, and so on, at exactly the
same IP address.

If a client is using "state-less" connections such as a standard browser
connection, then it would be unaware of a failover if it occurred between
server requests. If a failure occurs while a client is connected to the failed
resource, then the client will receive whatever standard notification is
provided by the client side of the application in use when the server side
becomes unavailable. This might be, for example, the standard "Abort, Retry,
or Cancel?" prompt you get when using Windows Explorer to download a file
at the time a server or network goes down. In this case, client reconnection
is not automatic (the user must choose "Retry"), but the user is fully
informed of what's happening and has a simple, well-understood method of
reestablishing contact with the server. Of course, in the meantime, MSCS is
busily restarting the service or application so that, when the user chooses
"Retry," it reappears as if it never went away.

For client-side applications that have "state-full" connections to the server, a
new logon is typically required following a server failure. In many cases, this
approach is required for security purposes. For example, this is how SAP R/3
works--if the server connection is lost, the user is prompted to log on again
to make sure it's the same user accessing the application.

Even with state-full connections, it's possible for an application to
automatically reconnect following a failover. For example, when Microsoft
demonstrated SAP R/3 failover at Microsoft Scalability Day, it was accessed
through an Active browser application that had automatically (and securely)
cached the user's ID and password from the initial logon. Thus, when the
server connection was momentarily lost during the failover demo, the client
application automatically logged on again using the cached ID and password.
This was done using standard IP connections, running a simple Microsoft
Visual Basic development system program within an HTML document through
the Microsoft ActiveX technology.

When a server comes back online following a failure, is there any human intervention required to get it back "up and running," or is the heartbeat enough for the other server to include it once again?

No manual intervention is required. When a server running Microsoft Cluster
Server, say "Server A," boots, it starts the MSCS service automatically. MSCS

in turn checks the interconnect (and network if necessary) to find the other server in its cluster, say "Server B." If Server A finds Server B, then Server A rejoins the cluster and Server B updates it with current cluster status info. Server A then initiates "failback," moving back failed-over workload from Server B to Server A at an appropriate time.

What is "failback," and how does it work in MSCS?
"Failback" is the ability to automatically rebalance the workload in a cluster when a failed server comes back online. This is a standard feature of MSCS. For example, say "Server A" has crashed and its workload failed-over to "Server B." When Server A reboots, it automatically finds Server B and rejoins the cluster. It then checks to see if any of the cluster groups running on Server B would "prefer" to be running on Server A. If so, it automatically moves those groups from Server B to Server A as soon as the time is right. Failback properties--that is, which groups can failback, which is their preferred server, and during what hours the time is "right" for failback--are all set from the cluster administration console.

Can the servers in an MSCS cluster be located at separate locations for recovery from site disasters?
Not at this time. All of the cluster configurations currently being considered for validation use SCSI connections to storage resources, which limits the distance between clustered servers to the distance supported by standard SCSI. This is typically no more than 25 meters, though there are SCSI extender technologies that can potentially stretch the connection up to 1,000 meters.

Note that Windows NT Server customers already have several choices for software that can mirror data to remote disaster recovery sites, including solutions from N.S.I., Octopus, Veritas, and Vinca. Most of these vendors have already announced that their disaster site mirroring solutions will also work with MSCS clusters.

Can MSCS restore registry keys for an application from one server to the other when doing failover?
Yes. Recovery of an application's registry information is a configurable feature that is available to the Generic Application and Generic Service resource types. Basically, you tell it what registry keys to log and recover, and that's all there is to it. This capability should be used if the application or service stores volatile information in specific registry keys. If this is done, when the resource comes online on another node, it will have the same registry information as the previously online resource.

When an application restarts on another server following a failure, does it re-start from a copy of the application?
No. The new server (say, "Server 2") would start the application from the same physical disks as Server 1, since ownership of the application's disks on the shared SCSI bus had been moved from Server 1 to Server 2 as one of the first steps in the failover process. This approach assures that the application always restarts from its last known state, as recorded on its disk drives (and, if you use the available option, as recorded in its registry keys.)

Can MSCS restore an application's "state" at the time of its failure rather than requiring a complete restart?
MSCS can restore the state of an application's registry keys, but any other

state information must be managed and restored by the application. Applications need to provide some model for persistence to insure that state can be recaptured. For example, Microsoft SQL Server™ uses transaction logs to provide this assurance. If a server running Microsoft SQL Server crashes, upon restart the application uses its transaction logs to bring the database back to a known state. With a cluster, just as with a single server, good application design and the use of ACID (Atomic, Consistent, Isolated, and Durable) transaction properties are important.

What is the granularity of resource failover?
MSCS supports failover of "virtual servers," which usually correspond to applications, Web sites, print queues, or file shares (including their disk spindles, files, IP addresses, and so on). MSCS also provides cluster-wide services that are simultaneously available on all servers in the cluster, including cluster administration, performance monitoring, event viewing, a cluster name, and cluster time synchronization.

What is a "quorum disk" and how does it help MSCS provide high availability?
It's a disk spindle that MSCS uses to determine whether or not another server is up or down. Technically, it's a resource that can only be owned by one server at a time, and for which servers can negotiate for ownership. Negotiating for the quorum drive allows MSCS to avoid "split brain" situations where both servers are active and think the other server is down. (This can happen when, for example, the cluster interconnect is lost and network response time is problematic.) The use of a quorum resource is one of the sophisticated algorithms that Microsoft got by working with pioneers in clustering such as Digital and Tandem.

Manageability   ▲ top of page

How does MSCS improve the manageability of servers?
MSCS gives administrators a graphical console from which they can monitor and manage all of the resources in a cluster as if it was a single system. Using the familiar standards of a Microsoft Windows graphical user interface, an administrator can use the cluster console to:

- audit the status of all servers and applications in the cluster.
- set up new applications, file shares, print queues, and so on, for high availability.
- administer the recovery policies for applications and resources.
- take applications offline, bring them back online, and move them from one server to another.

The ability to graphically move workload from one server to another with only a momentary pause in service (typically less than a minute) means administrators can easily unload servers for planned maintenance without taking important data and applications offline for long periods of time.

Does MSCS provide administrators with a "single system image"?
Yes. MSCS provides administrators a single graphical console to manage all of the applications and resources in a cluster. The MSCS console presents cluster resources by physical server, and by "virtual server" (or "cluster

group"). This allows administrators to centrally manage the cluster as a collection of virtual application-oriented servers, or as a collection of physical resources when appropriate.

Can MSCS be remotely managed?
Yes. An authorized user can run the MSCS administration console from any Windows NT Workstation or Windows NT Server on the network.

How does MSCS help administrators do "rolling upgrades" of their servers?
With MSCS, server administrators no longer have to do all their maintenance within those rare windows of opportunity when no users are online. Instead, they can simply wait until a convenient off-peak time when one of the servers in the cluster has enough horsepower for all of the cluster workload. They then point-and-click to move all the workload onto one server, and they're ready to perform maintenance on the unloaded server. Once the maintenance is complete and tested, they bring that server back online and it automatically rejoins the cluster, ready for work. When convenient, the administrator repeats the process to perform maintenance on the other server in the cluster. This ability to keep applications and data online while performing server maintenance is often referred to as doing "rolling upgrades" to your servers.

Will Microsoft support "rolling upgrades" of future server products using MSCS clusters?
It is Microsoft's goal to support "rolling upgrades" between releases of Microsoft server software using MSCS clusters. However, we cannot commit to this for all releases of all products. Persistent storage formats must occasionally change to accommodate new capabilities, and changes in persistent storage occasionally require applications to be taken offline while storage or indices are restructured. Microsoft will commit to always providing smooth upgrades between releases of all our products, and we'll use MSCS to provide seamless rolling upgrades whenever possible.

Scalability   ▲ top of page

How will MSCS enhance server scalability?
The manageability benefits of the initial version of MSCS will simplify many of the processes currently used to improve scalability, such as upgrading server hardware and installing new versions of applications.

There are scalability advantages to clustering, for example:

- For file sharing and print services, some of these shares/queues can be provided by one node and some by the second node. IIS services can be provided by both nodes simultaneously.
- For branch office automation, the total workload can be partitioned so that Exchange Enterprise Edition 5.5, for example, runs on one node, but print, file and web services are provided by the second node.
- For database applications, the OLTP SQL Server Enterprise Edition 6.5 database can be on one node and the data warehouse on the other node.
- For a three tier application, the application/business logic can run on one node and the database on the other node.

Writing Microsoft Cluster Server (MSCS) Resource DLLs

Microsoft Corporation

1997

Page Options

Average rating:
8 out of 9

✎ Rate this page

🖨 Print this page

✉ E-mail this page

★ Add to Favorites

Abstract

Microsoft® Cluster Server (MSCS) allows multiple Microsoft Windows NT® operating system-based servers to be connected together, making them appear to network clients as a single, highly available system. This paper provides a high-level overview of the processes involved in writing well-behaved cluster applications for MSCS. This document also describes how application and service developers can take full advantage of MSCS by writing resource dynamic-link libraries (DLLs), debugging their applications, and installing their applications and services in a cluster environment.

Contents

Introduction

Microsoft Cluster Server (MSCS) allows multiple Microsoft Windows NT operating system-based servers to be connected together, making them appear to network clients as a single, highly available system. From the system administrator's viewpoint, MSCS provides the additional advantage of easy administration and scalability, and the MSCS architecture provides a standard infrastructure for scalable, cluster-aware applications in future versions.

The purpose of this document is to provide a high-level overview of the processes involved in writing well-behaved applications that can take advantage of the Microsoft Cluster Server capabilities. This document describes how you can take full advantage of MSCS by writing resource dynamic-link libraries (DLLs), debugging your applications and services, and then installing them in a cluster environment.

> **Note** This White Paper assumes that you have successfully installed the Microsoft Cluster Server software in a cluster environment and also have the Microsoft Platform Software Development Kit (SDK) and the build environment working. If you are having trouble setting up the development environment, please refer to your Platform SDK documentation.

Clustering and High Availability

Microsoft Cluster Server allows applications and services to run more efficiently on Windows NT Server by directing client requests based on resource availability and server load. (In the first release of MSCS, load balancing is done manually; future releases will

IPAGE 25/28 * RCVD AT 2/13/2004 3:00:32 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/1 * DNIS:8729306 * CSID:215 * DURATION (mm-ss):08-10

1/29/200

provide automatic load balancing.) If one of the systems—or *nodes*—in the cluster is unavailable or has failed due to hardware or software problems, its workload is handled by other systems in the cluster until the failed systems are brought back online.

Note that Microsoft Cluster Server is designed to provide high availability, rather than true fault tolerance. The phrase "fault tolerant" is generally used to describe technology that offers a higher level of resilience and recovery. Fault-tolerant servers typically use a high degree of hardware or data redundancy, combined with specialized software, to provide near-instantaneous recovery from any single hardware or software fault. These solutions cost significantly more than a clustering solution because you must pay for redundant hardware that waits idly for a fault from which to recover. Microsoft Cluster Server provides a very good high-availability solution using standard, inexpensive hardware, while maximizing computing resources.

The Shared-Nothing Model

Microsoft Cluster Server version 1.0 is a two-node cluster that is based on the *shared-nothing* clustering model. The shared-nothing model dictates that while several nodes in the cluster may have access to a device or resource, the resource is owned and managed by only one system at a time. (In an MSCS cluster, a resource is defined as any physical or logical component that can be brought online and taken offline, managed in a cluster, hosted by only one node at a time, and moved between nodes.)

Each node has its own memory, system disk, operating system, and subset of the cluster's resources. If a node fails, the other node takes ownership of the failed node's resources (this process is known as *failover*). Microsoft Cluster Server then registers the network address for the resource on the new node so that client traffic is routed to the system that is available and now owns the resource. When the failed resource is later brought back online, MSCS can be configured to redistribute resources and client requests appropriately (this process is known as *failback*).

> **Note** When a node fails, any clients are disconnected. For the failover to be truly transparent, client applications must be written to reconnect in the event of node failure.

A generic MSCS cluster setup is shown in Figure 1, below.
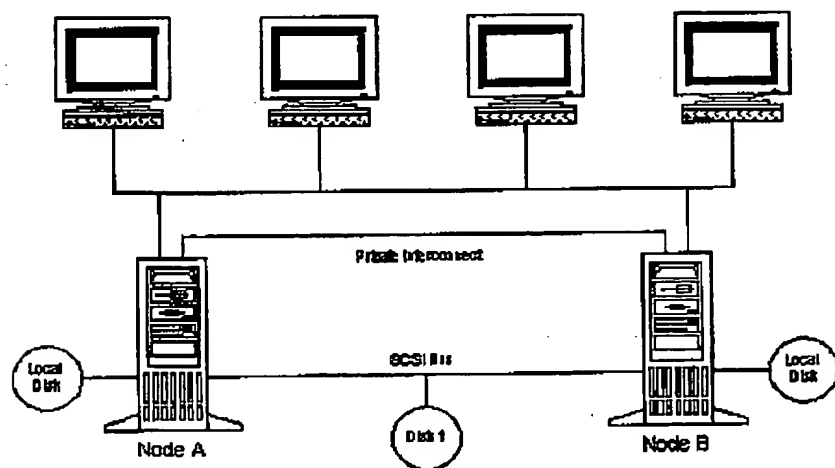
Figure 1. Standard two-node MSCS configuration

The following section provides an introduction to MSCS architecture.

PAGE 26/28 * RCVD AT 2/13/2004 3:00:32 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/1 * DNIS:8729306 * CSID:215 * DURATION (mm-ss):08-10

1/29/200

Microsoft Cluster Server Architecture

Microsoft Cluster Server is comprised of three key components:

- The Cluster Service
- The Resource Monitor
- Resource and Cluster Administrator extension DLLs

The Cluster Service

The Cluster Service (which is composed of the Event Processor, the Failover Manager/Resource Manager, the Global Update Manager, and so forth) is the core component of MSCS and runs as a high-priority system service. The Cluster Service controls cluster activities and performs such tasks as coordinating event notification, facilitating communication between cluster components, handling failover operations, and managing the configuration. Each cluster node runs its own Cluster Service.

The Resource Monitor

The Resource Monitor is an interface between the Cluster Service and the cluster resources, and runs as an independent process. The Cluster Service uses the Resource Monitor to communicate with the resource DLLs. The DLL handles all communication with the resource, thus shielding the Cluster Service from resources that misbehave or stop functioning. Multiple copies of the Resource Monitor can be running on a single node, thereby providing a means by which unpredictable resources can be isolated from other resources.
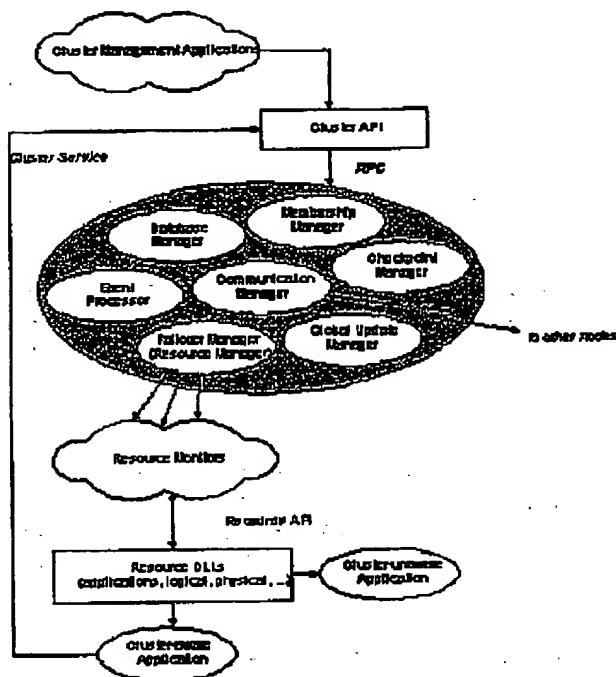
The Resource DLL

The third key Microsoft Cluster Server component is the resource DLL. The Resource Monitor and resource DLL communicate using the Resource API, which is a collection of entry points, callback functions, and related structures and macros used to manage resources. Applications that implement their own resource DLLs to communicate with the Cluster Service and that use the Cluster API to request and update cluster information are defined as *cluster-aware applications*. Applications and services that do not use the Cluster or Resource APIs and cluster control code functions are unaware of clustering and have no knowledge that MSCS is running. These cluster-unaware applications are generally managed as generic applications or services.

Both cluster-aware and cluster-unaware applications run on a cluster node and can be managed as cluster resources. However, only cluster-aware applications can take advantage of features offered by Cluster Server through the Cluster API. For example, cluster-aware applications can:

- Report status upon request to the Resource Monitor.
- Respond to requests to be brought online or to be taken offline gracefully.
- Respond more accurately to **IsAlive** and **LooksAlive** requests.

MSCS includes two tools that perform basic cluster management: these tools are Cluster Administrator (CluAdmin.exe) and a command-line management tool (Cluster.exe). You are encouraged to write your own custom management tools if needed; however, any lengthy discussion of managing cluster-unaware applications or developing cluster management tools is beyond the scope of this paper.

Figure 2 shows how the Cluster Service, Resource Monitor, and resource DLLs interact with each other on a single node running the Windows NT Server, Enterprise Edition operating system, cluster management applications, and both cluster-aware and cluster-unaware applications.

PAGE 27/28 * RCVD AT 2/13/2004 3:00:32 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/1 * DNIS:8729306 * CSID:215 * DURATION (mm-ss):08-10

1/29/20(

Figure 2. MSCS components on a single node running Windows NT Server

Note that cluster-aware applications should also implement Cluster Administrator extension DLLs, which contain implementations of interfaces from the Cluster Administrator extension API. A Cluster Administrator extension DLL allows an application to be configured into the Cluster Administrator tool (CluAdmin.exe). Implementing custom resource and Cluster Administrator extension DLLs allows for specialized management of the application and its related resources, and enables the system administrator to install and configure the application more easily.

Next, this paper describes resources and resource DLLs in greater detail, and describes the reasons for writing a custom resource DLL.

Resources and Resource DLLs

To the Cluster Service, a resource is any physical or logical component that can be managed. Examples of resources are disks, network names, IP addresses, databases, Web sites, application programs, and any other entity that can be brought online and taken offline. Resources are organized by *type*. Resource types include physical hardware (such as disk drives) and logical items (such as IP addresses, file shares, and generic applications).

Every resource uses a resource DLL, a largely passive translation layer between the Resource Monitor and the resource. The Resource Monitor calls the entry point functions of the resource DLL to check the status of the resource and to bring the resource online and offline. The resource DLL is responsible for communicating with its resource through any convenient IPC mechanism to implement these methods.

Note that applications or services that do not provide their own resource DLLs can still be configured into the cluster environment—

PAGE 28/28 * RCVD AT 2/13/2004 3:00:32 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/1 * DNIS:8729306 * CSID:215 * DURATION (mm-ss):08-10

1/29/20(